

Development of Vidivox

SOFTENG 206 Project

Ammar Bagasrawala

Executive Summary

Vidivox is an application created as part of the SOFTENG 206 course with the purpose of allowing users to manipulate audio and video files. Clients were approached concerning the main requirements of the application and it was concluded that the basic functionality included the means to watch a video and overlay a video with a user specified audio track. This audio track was also to be able to be generated through the use of Vidivox, by allowing the user to type text into the application and by allowing the user to select an mp3 file to overlay the video with. The Vidivox application was to also contain extra functionality to serve as bonus features. This version of Vidivox included the ability to manipulate video effects such as the brightness, hue, saturation, gamma and contrast levels. Other features which extended from the main requirements were the manipulation of audio features. This included the ability to generate audio from the text input with different accents and different levels of pitch. The accents include a make robotic voice, a male British voice and a male Kiwi voice. This report discusses the final product Vidivox, its beta version and the development of this application. Such discussions encompass the user interface aspect, functionality, the software architecture and the evaluation of the application. The results of the evaluation highlight the aspects in which the beta version lacked, so as to enable the final version to be of a higher quality. Although, due to resource constraints such as time, further work was unable to be carried out on the application and hence all future development has also been outlined in this report.

Table of Contents

1.	Introduction.....	4
1	Graphical User Interface Design.....	5
1.1	Language and Package Choices	5
1.2	GUI Layout and Presented Information Decisions	5
1.3	Colour Decisions.....	6
2	Vidivox Functionality	7
3	Vidivox Development and Code Design	9
3.1	Language and Library Decisions	9
3.2	Software Architecture	9
3.3	Development Process.....	9
3.4	Innovations in Implementation	10
4	Evaluation and Testing	11
4.1	Self-Testing and Evaluation of Vidivox	11
4.2	Peer Evaluation Results	11
5	Conclusion	13
5.1	Future Improvements	13
5.2	Overall Conclusions.....	13
6	References.....	13

1. Introduction

The needs for multimedia manipulation have increased over the past few decades. From the early stages of just being able to view videos to applications which let users create their own videos. The Vidivox application fits into the category as an application which provides the means for multimedia manipulation. It allows users to merge videos and synthesised audio tracks with each other into a single video file. The users can specify an external audio source, such as an mp3 file, or write in some text which is then converted into an audio file by the application. Audio created from the input of text by the user also has options for how the audio should be manipulated. These effects on the audio include the choice of three accents (robotic, British and Kiwi) and the choice of three different pitches (low, normal, high). Vidivox also allows for the manipulation of the video itself through video effects such as change in the brightness, hue, contrast, saturation and gamma levels of the video.

These are the essential functions of the Vidivox application as outlined by the requirements set forth by the clients. These clients, as part of the intended users were met with on a weekly basis to consult the features and functionalities of the application. These design meetings proved to be highly useful as they provided a clearer understanding of what the end product should look like and the functionality it should contain. One of the key requirements was a user-friendly application which was intuitive in its use and enabled a clear understanding of its features. This was because the intended audience and users were those not familiar with the idea of video and audio manipulation. In other words, the application was to be designed for those who have spent very little time working with multiple video and audio files.

This report discusses the development of the Vidivox application and the decisions that were made throughout its development lifecycle. These include discussions on the user interface and the functionality of the product, along with the evaluation of the beta and final applications.

1 Graphical User Interface Design

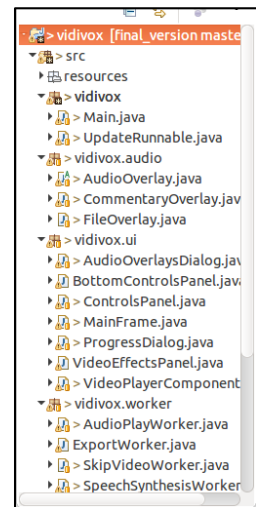
1.1 Language and Package Choices

The Vidivox application was designed with the aspect of ease of use in mind. Effort was put into the application's graphical user interface to ensure that the user was able to navigate through Vidivox and use its provided functionality in the most effective way. However, in order to provide a neat and user-friendly interface, the code behind the application would need to be constructed in a well-structured manner. This meant the effective use of classes and packages to separate segments of the functionality and UI so that cohesion was improved. The Java language was therefore implemented to achieve high cohesion as it allowed for the separation of code into methods, classes and packages.

The packages used in the creation of Vidivox include:

- **Vidivox** – The main package which contains the classes associated with starting the application and keeping it running. It is also the package which contains all other packages
- **Ui** – The package which contains all the classes that are associated with the creation and handling of the user interface.
- **Audio** – The package in which the classes are used to handle the audio creation and the GUI when audio is to be added to the video.
- **Worker** – This package is where all the classes that extend `SwingWorker` are placed so that the functionality which causes background processes to spawn can all be associated to this one package.

The use of these packages enables the code to be structured in a sensible manner and thus increases the robustness and readability of the code.



1.2 GUI Layout and Presented Information Decisions

The amount of information being presented to the user at once was also a major factor concerning the success of the GUI of Vidivox. With these factors in consideration, the GUI was designed with a minimalistic approach so as to not overload the user with information at any given time. Hence the only information presented to the user upon the starting of the application is what is shown by the

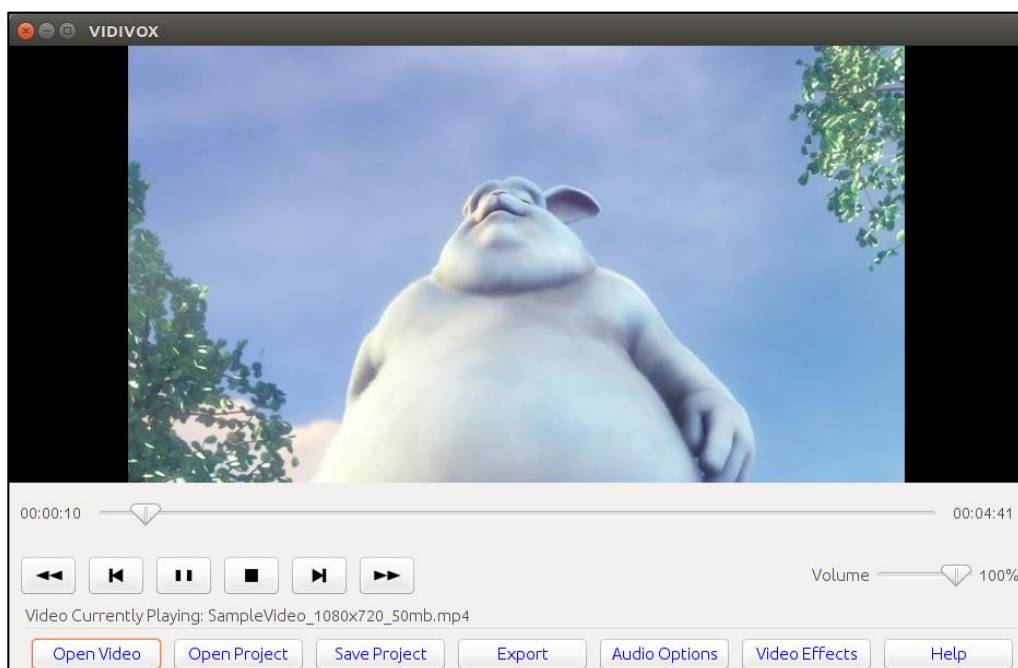
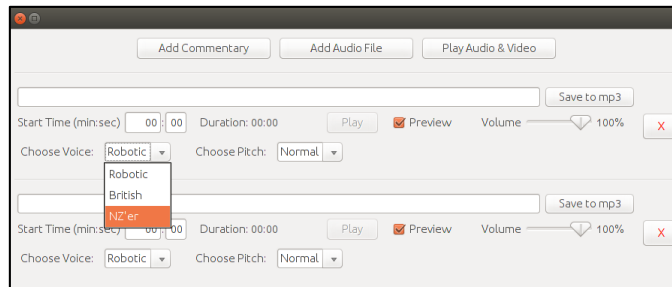


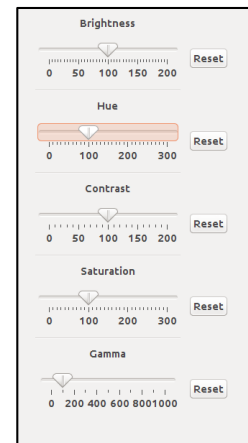
image above. It provides a limited number of options so that the user has the time to take in all the information and understand the GUI without being overwhelmed. The layout of the buttons and other components such as labels was also considered during the development process. As such, most of the GUI was laid out through the use of the GridBagLayout to ensure the spacing between the components was not changed when the user resized the frames. Thus ensuring that the readability of the information in the GUI was not decreased.



Keeping with the mind-set of having as little information showing at once as possible while still providing enough to make the application user-friendly. The audio frame, where the commentary and other audio can be added to the video, has panels instantiated only when the user decides to add more audio. For example, the image above shows the audio frame

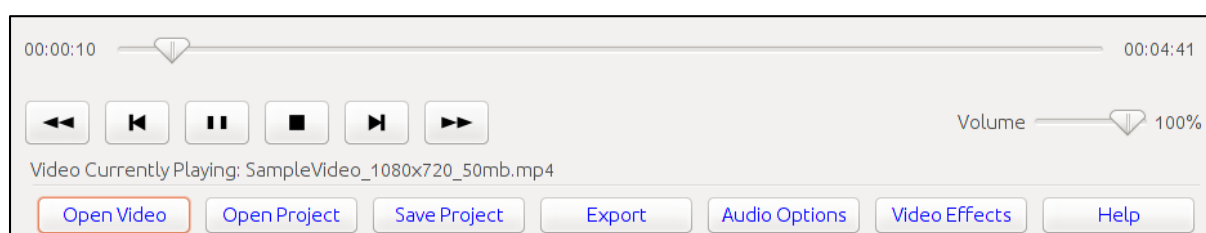
with two commentary panels, if the user would click the “Add Commentary” button, another panel would be created under these two. This decreases the chance of the user being overwhelmed by the application and thus makes it more user-friendly. The same goes for the video effects panel, it is hidden unless the user decides they want it to be seen. The panel is then extended from the right side of the main frame.

Choice of components to present certain information was also a substantial factor in determining the success of the GUI. For example, a JSlider was used to represent the progress of the video as it was playing. JSliders were also used to control the volume and the video effects such as hue, brightness etc. JSeperators were another component used to enhance the visual effect and readability of the GUI. Examples of this being used can be seen in the video effects panel, each of the effects (brightness, contrast, hue etc.) are sectioned off through the use of the JSeperators to enable the user to understand which of the sliders and reset buttons corresponded to a certain video effect.



1.3 Colour Decisions

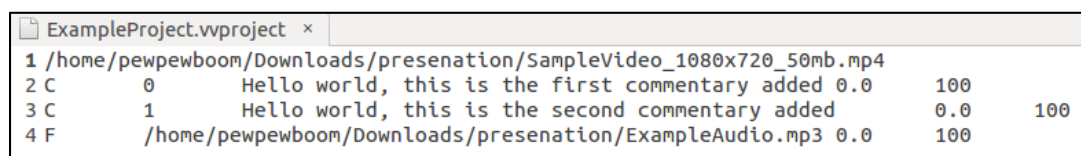
Colour was another factor considered during the development of Vidivox. As colour has the ability to increase or decrease the usability and understanding of the user interface. In Vidivox, the colours used include blue, red, black, white, orange and grey. Where the most used colours are grey and black. This was because this pair of colours allows the information to be presented clearly and in a non-overwhelming manner for the users. The colours blue, red, and orange were used to highlight certain implications of the buttons, for example red was used for the delete audio button as is the norm with delete buttons in other applications. The blue was used only in the main frame in order to separate those buttons with the others (which were black) to show the user that the functionality of buttons in blue was not directly related to the video being played.



2 Vidivox Functionality

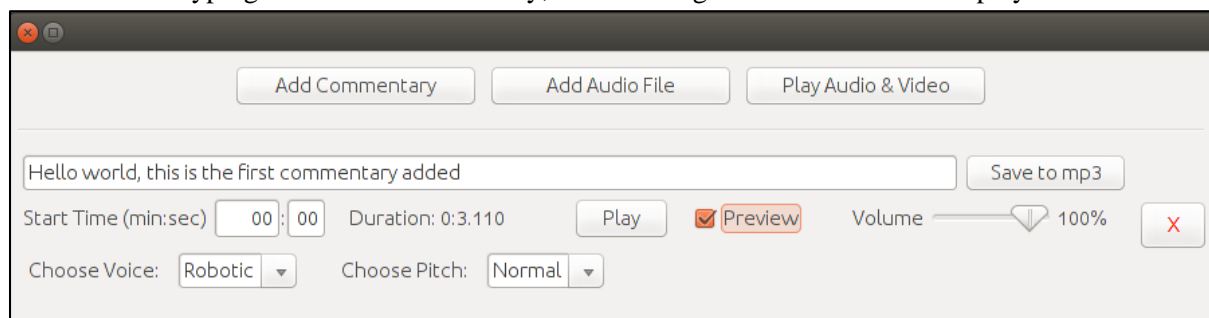
Vidivox, an application based on the Java framework for VLC, allows for multiple functions. These include the basic ability to view videos in mp4 or avi format in a user-friendly manner. However, Vidivox also allows for the modification of a video file. Such as the change of brightness, hue, contrast, saturation and gamma whilst the video is being played. The audio of the video can also be overlapped with other audio (generated externally or within Vidivox itself) and then exported into a single video file.

Throughout the development cycle of Vidivox, decisions were made based on the functionality the application should contain and how the selected functionality should be carried out. These decisions encompassed the requirements of the clients so as to provide them with an application they deemed appropriate and fit for purpose. This meant that after each of the design meetings with the clients, the application was examined to determine whether it complied with the requirements. The design decisions were then made based on the results of this examination. For example, at a particular design meeting, the clients were approached with the idea of a character limit when text was entered to be converted into audio. At that stage in development, the application did not have any limit on the number of characters a user could enter. This was addressed after the decision to add a 100 character limit on a single commentary was made.



1	/home/pewpewboom/Downloads/presentation/SampleVideo_1080x720_50mb.mp4
2	C 0 Hello world, this is the first commentary added 0.0 100
3	C 1 Hello world, this is the second commentary added 0.0 100
4	F /home/pewpewboom/Downloads/presentation/ExampleAudio.mp3 0.0 100

There were many major decisions in the development lifecycle. One of these was in the early stages of development, where the basic layout of the application and the manner in which the user would navigate through the application was to be determined. After consideration of the target users, (inexperienced video editors) it was decided that the Vidivox application would be designed with a minimalistic approach. This meant the instantiation of the JPanel containing the options to add commentary whenever the “Add Commentary” button was clicked. Examples of another decision made throughout the Vidivox development included the “Open Project” and “Save Project” features. This was deemed necessary as the user would find the application much more usable if they were able to save all the temporary changes they made to a particular video file and then come back and modify it further. The project files themselves included only the information of the path to the video file being modified, and the information on the commentaries or audio files added to the project as can be seen by the image below. This added functionality would also allow the user to preview the final video that they would be able to export which merges the audio tracks and video together into one mp4 video. Another major usability decision made was the addition of the preview feature. This allowed the user to watch the video as if it were already exported. The preview generated the wav file once the user had finished typing in their commentary, and through the use of the ffmpeg tool and the



EmbeddedMediaPlayerComponent class's methods, multiple audio tracks and the video were able to be played immediately without the need of exporting. This function was costly in terms of time, however it was deemed necessary and thus the decision to implement it was made. Additional voice features were also implemented for the user to work with. These included the option to change the type of voice to one of three options – robotic, British or Kiwi. The other voice feature was to change the pitch of the voice to either low, high or normal. These changes to the voice were implemented through the use of a scheme file which served as an argument to the text2wav function. The scheme file contained several commands which were passed to the festival tool installed on the computers to alter the audio created.

However, there was a limit to the amount of functionality provided to the user so as to prevent from overwhelming them. This meant functionality such as changing the speed of the voice was not included in the final version of Vidivox. This provided the user with a more easy to use application rather than one which contained numerous functions but proved difficult to use.

3 Vidivox Development and Code Design

3.1 Language and Library Decisions

The development of Vidivox consisted of using the Java language to design and implement the functionality of the application. This was because Java is a multi-platform, object orientated language which enabled the use of high quality software architecture during development. The Vidivox application is designed for use on Linux systems, as key parts of the functionality of the application included the use of Festival, which is a tool for text to speech conversion on Linux systems. The other library that was used in the development of Vidivox, was the VLCJ library which is a Java based framework for the VLC media player. The fact that Java is an object orientated language allows for the implementation of fundamental aspects of software architecture such as packages, classes and methods. Which means that an application produced through the use of good coding structure enabled it to be robust, usable and of a high quality. Java also consisted of the Swing library, which enabled the creation of user interfaces and was used for the development of this applications GUI. The Swing library also already consisted of methods which was ideal in providing time effective solutions to problems encountered during the development of the GUI and its functionality. Another Java library used for GUI creation was considered, namely Java FX. However, with a prototype already created with the use of Swing and because of the limits of time, it was decided that Swing would be a better option. Therefore due to these reasons, it was decided that Java would be a suitable language to create Vidivox in compared to other languages such as C#, JavaScript etc. However, it wasn't the only language used. Bash commands were also used for certain parts of the application. For example, the bash command `text2wav` was used to convert user entered text into a wav file. These commands were incorporated into the Java methods and classes as an extension of `SwingWorker` (so as to multi-thread processes and avoid the GUI freezing).

3.2 Software Architecture

Vidivox's software architecture is structured in a manner which differentiated classes that provided certain functionalities from each other into separate packages. For example, the classes associated with the creation of the user interface were placed into one package so as to increase the readability and understanding of the code. By doing this, the robustness of the produced application increases, and also decreases the difficulty in finding and fixing faults as the programmer would be able to recognize which package is causing errors, and reduces the amount of code that needs to be looked at. Due to this logical structure of separating segments of code based on the functionality they provide, it would seem that even if the development process were to begin anew, the same code structure would be implemented.

3.3 Development Process

The development process for the Vidivox application was a simple one, as it was a small scale project which at most involved two people working on a single application during the development phase. It could be defined as following a small scale version of the Waterfall Model with backflow. As basic requirements were first set out, the design phase was then carried out, followed out by an implementation phase and finally the testing phase. However, in the development of Vidivox, these phases often occurred at the same time as the requirements were frequently updated after client meetings. Due to this, design and implementation did become less efficient as it would be discovered that a certain function needed to be implemented, and was not required in the initial stages of development. Which meant that decisions made during the implementation and design phases needed to be re-evaluated, thus meaning that code needed to be re-written to encompass the new functions.

As this was a large project for a small team (two people at most), the version of code had to be controlled and concurrency of files being shared between team members also had to be managed. The

Version control system applied to the development of Vidivox was Git, which served as an online repository that could be accessed through GitHub. The use of this repository enabled a smoother workflow throughout the development and also ensured that files were backed up on a regular basis. Git allowed both of the team members to access files at the same time and thus saved time for both members during the implementation phase of development. After changes

Commits on Sep 25, 2015		
fixed exporting sometimes not working properly when more than 1 audio...	ChineseElectricPanda committed on Sep 25	81f1319
fixed commands not working when filepaths contain spaces	ChineseElectricPanda committed on Sep 25	11cb9b4
Added readme files	ammar27 committed on Sep 25	94e9c93
Merge branch 'master' of github.com:ChineseElectricPanda/vidivox.git	ammar27 committed on Sep 25	2F3d8b3
Updated the file choosers so only certain file types can be opened	ammar27 committed on Sep 25	4b141fe
Merge branch 'master' of github.com:ChineseElectricPanda/vidivox	ChineseElectricPanda committed on Sep 25	a393c22
fixed audio and slider behaviour	ChineseElectricPanda committed on Sep 25	257f8c6
Merge branch 'master' of github.com:ChineseElectricPanda/vidivox.git	ammar27 committed on Sep 25	97b6727
Fixed bug where audio doesn't stop playing once video ends	ammar27 committed on Sep 25	0f5ecfe
Merge branch 'master' of github.com:ChineseElectricPanda/vidivox	ChineseElectricPanda committed on Sep 25	0757734

were made and the build was tested so as to ensure it was not broken, the latest changes would be “pushed” to the online repository for both members to access. This was done on a regular basis so as to avoid major changes as it would result in more errors while merging different versions which did become an issue during the early stages of development as the GUI implementations took up lots of space and resulted in many textual merge errors. Branches were also created and pushed to when certain functionality was implemented but not fully tested. The creation of test branches in the repository meant that if the code that was last pushed to it, broke the build, then only that branch would be broken and the main branch would not have been affected. Thus acting as a safety net for code that was not fully tested or fully functional. The use of Git enabled the development process to be safer and allowed for a smoother workflow because of its clear and easy to use features.

3.4 Innovations in Implementation

In accordance with the basic functionality required of Vidivox by the clients there were several innovations added to the application. The basic functionality includes the ability to view a video file, add audio at specific times to the video through either an mp3 file or some text specified by the user. The ability to preview the audio alone and export the audio tracks and video as one video file are also basic functions specified by the clients. However, Vidivox also incorporates the idea of projects into its key functions. Saving a project stores the path to the video and added audio files, the commentary added by the user is also stored in this text file which is set to end in “.vvproject”. When this project file is opened by Vidivox, it reads the contents of the file and adds the components to the GUI accordingly. This enables the user to save their video editing progress and work on it at a later date. Another innovation implemented in Vidivox, is the modification of the video, such as the brightness, hue, contrast, saturation and gamma of the video. This was achieved through the use of the VLCJ library which contained easy to implement methods such as “setHue()”. Yet another innovation was the ability to preview the video and audio tracks immediately. Using background processes to run ffmpeg and text2wav, the wav and mp3 files were created while the user was typing the commentary into the text box. This meant that as soon as the user finished typing, a wav file would be ready to play using ffmpeg if the user wanted to preview the exported video file. Using a mouse listener for the video surface, a feature was implemented where if the user clicks on the video, it would pause or play depending on its current state. It was an innovation inspired by modern day video players such as that used in YouTube. During the development, it felt that the voice related options were not enough, and thus a feature was added to modify the voice with the use of three different accents (robotic, British, Kiwi), and three different levels of pitch (low, normal, high). All these innovations in accordance with the basic features, enhance the Vidivox application.

4 Evaluation and Testing

4.1 Self-Testing and Evaluation of Vidivox

Throughout the development of Vidivox, the need for testing and an evaluation on the GUI and its related functionality arose. However, due to limitations on human resources such as the client and their time, the application could not be tested extensively by them. This meant that Vidivox, during most of the development process was tested by an individual (myself or my partner).

The testing was carried out by running the application and while being in the mind-set of the target user (non-professional video editors), the features in the application were used. If an error occurred, the aim was to find where the fault in the code occurred. This was made easier through the use of the IDE Eclipse's error messages which presented themselves on the console and enabled quick access to locations of errors which threw exceptions. An example of such an error was a constant null pointer exception being thrown when a `SwingWorker` class was trying to kill a process. It was solved through the use of an if statement to check if the process existed. However, these errors were not too difficult to locate. Other errors, such as those related to the user interface were harder to locate and repair. For example, an error was encountered where if the user opened a project, and tried to add more commentary or audio to it, Vidivox would not let it. It was only after creating many instances where the code printed values of variables on the console, was the fault found. A missing call to a particular audio class and its method to set up the listeners was causing the error of disabling the functionality of the buttons and thus not updating the GUI as it was supposed to. These types of faults were more time consuming to fix compared to those created by logical errors, however there were more logical errors encountered throughout the development process.

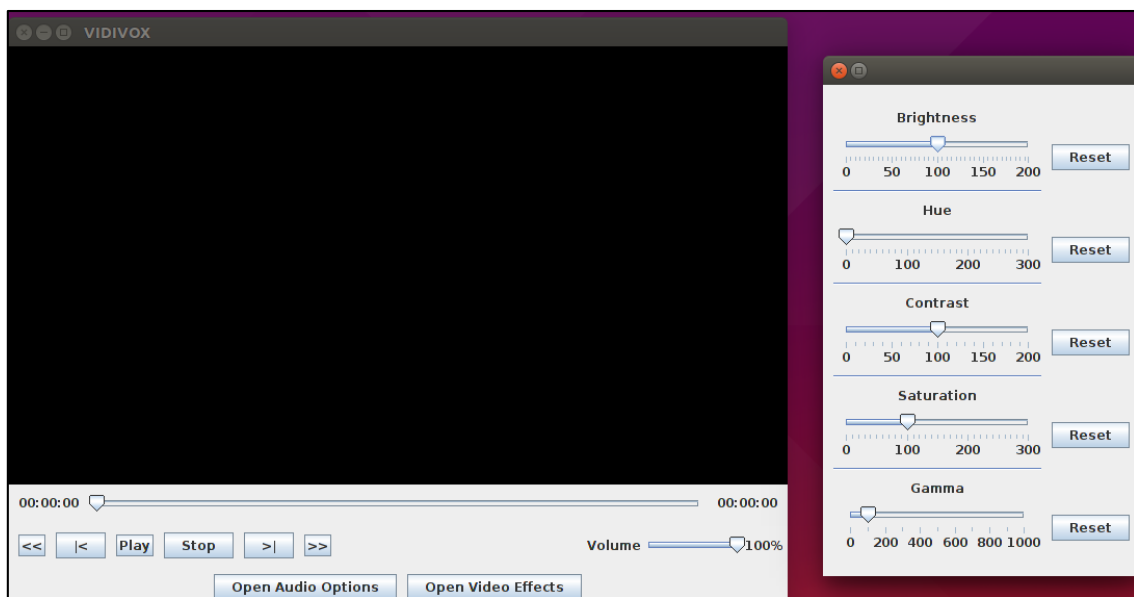
4.2 Peer Evaluation Results

At a particular point in time, October 12th to be precise, a beta version of Vidivox was submitted to an online system where peers working on the same project would be required to test each other's Vidivox applications and evaluate them. The results from this peer evaluation was taken into consideration for the final version of Vidivox.

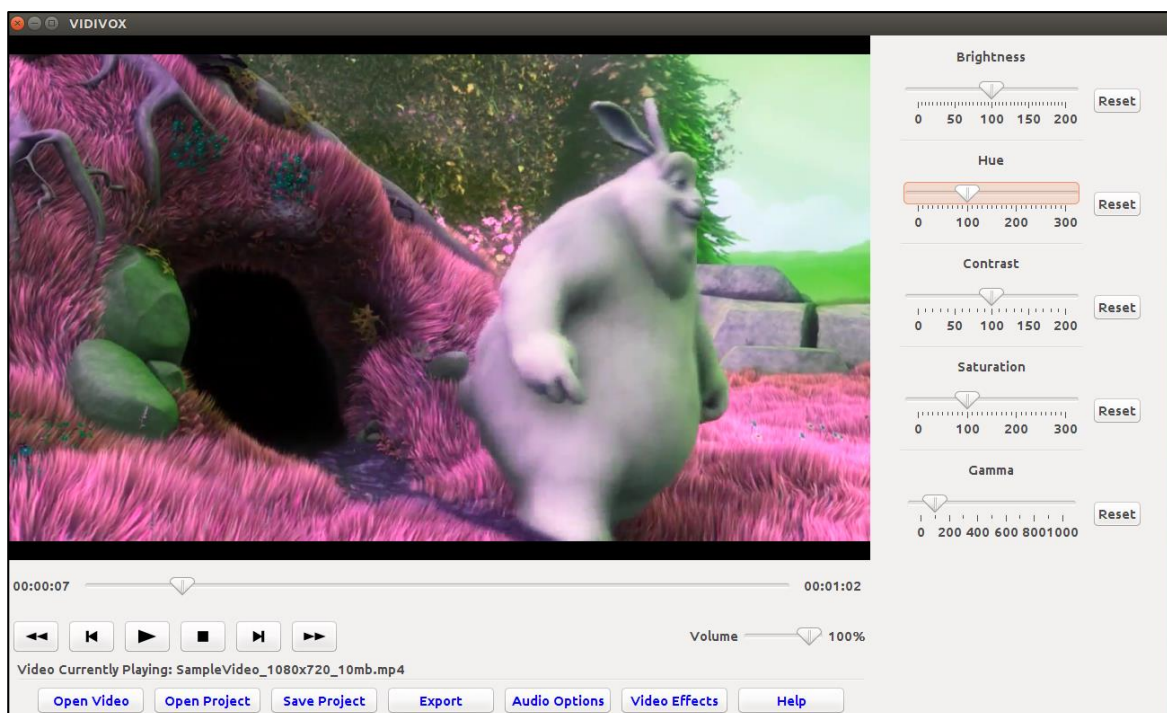
From the peer evaluations, it can be concluded that most of the reviewers found the beta version easy to use and quite intuitive. However there were comments which showed that the beta did need improving in certain departments. For example a peer commented that symbols should be used for certain buttons in the main frame such as the play, pause and stop buttons in order to increase the visual appeal of the GUI. Several comments also pointed out the fact that the application created more than one frame when the corresponding buttons were clicked. These frames include the dialogs for adding commentary and audio, and the dialog for the video effects. These two windows in accordance with the main frame, essentially caused the usability of the application to decrease as the users found that they had to keep switching between these windows and this proved to be a nuisance. The only bug found by those reviewing this beta version of Vidivox, was the ability for the user to create many windows for the addition of commentary and video effects.

The results from this evaluation provided insight on how to further develop the application and were taken into consideration for the final version of Vidivox. The bug where many windows could be opened was the first to be addressed, as it was the simplest to fix. Once it was certain that all faults were taken care of, the alteration of the GUI design was considered. Through the use of in built functions such as "`UIManager.setLookAndFeel()`", the application was given a new style of Java components as can be seen by the figures below. Along with this, the addition of images to buttons such as play, pause etc. were also added. These changes addressed the beautification issue a peer raised. Another minor issue raised was the lack of voice alteration features such as change of speed, pitch and accent. In order to satisfy the clients and the peers, two of these suggested voice alterations

were taken into consideration and implemented, namely the choice of three accents and the choice of three pitch levels. Speed of the audio was not implemented as it proved to disrupt the layout in the GUI by making the currently existing elements squashed and not aligned well. The problem where users did not know how to implement certain functions such as export and previewing the audio with the video was also addressed. This was achieved by adding tool tip text and a help button in the main frame, this would allow the user to clearly identify the functions of each of the components in the GUI. Most of the suggestions for improvement were taken into consideration so as to provide a great user and easy user experience in relation to Vidivox.



Beta Version



Final version

5 Conclusion

5.1 Future Improvements

Although the current version of Vidivox allows the functionality to be conveyed in a logical and structured manner, there could definitely be further improvements to the GUI design. As some aspects of the functionality such as adding audio or modifying the video through video effects, are not conveyed as effectively as possible. For example, both the audio and video effects could be added to the mainframe instead of just the video effects. As this would avoid the need for more than one window to be open when using Vidivox, thus increasing its usability aspect. Extra features in terms of audio manipulation could also be added, one such example is the addition of the audio speed manipulation. Another feature that would improve the use of Vidivox, is the addition of a timeline which would enable the user to graphically see where their added audio plays in relation to the video. However, in accordance with addition of extra features, many of the existing features still need to be enhanced. For example, the warnings that pop-up when the user makes an invalid choice or doesn't correctly use a function could be improved so as to inform the user in a more effective way (use of the colour red, bolded text etc.). The video effects in the current version of Vidivox also needs improvement as it does not save the effects made to the video, it only allows the user to make changes while playing the video. Another function which was room for improvement is the enhancement of the help button. In future versions, the help button could enable the user to select whether they want to read the manual or not, and open up the manual if they choose to read it. All these additions and enhancements of existing features would increase the quality of the Vidivox application greatly, making it more user friendly, intuitive and robust.

5.2 Overall Conclusions

Through the development of this product Vidivox, many lessons were learnt. This included lessons on version control, where the use of Git enabled team members to share files in a seamless manner. Many new features of Java were also discovered such as the VLCJ library which allowed for manipulation of videos. Concurrency was also a large aspect of the development of Vidivox, and the use of worker threads and background processes to accomplish certain functions enabled for a more in depth understanding behind concurrency. Another major lesson learnt, was the ability to work in a team on a software project. This would be a significant factor in determining the success of future projects and is considered a fundamental part of the Vidivox project. Overall, it can be concluded that the end product, Vidivox, was created with customer usability in mind, and with the intentions to provide a robust application for the manipulation of audio and video files.

6 References